AD-A041 039 DBA SYSTEMS INC MELBOURNE FLA F/G 12/1

DIGITAL TERRAIN DATA COMPACTION USING ARRAY ALGEBRA.(U)

NOV 76 U RAUHALA, S GERIG DAAG53-76-C-0127

UNCLASSIFIED ETL-0108 NL

| OF |

AD

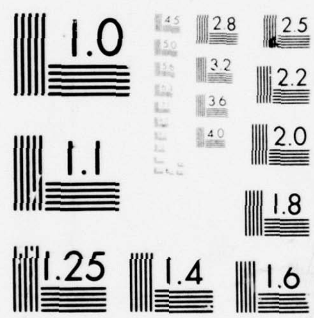
A041039

END

DATE

FILMED

7 - 77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

ETL-0108

AD A041039

DBA SYSTEMS, INC.

Photogrammetry

Computer Services

Mathematical Research

Electro-Optical Instrumentation

Approved For Public Release
Distribution Unlimited

GENERAL OFFICES
Post Office Drawer 550
Melbourne, Florida 32901
Phone: 305 727-0660

ELECTRO-OPTICAL DIV.
Post Office Drawer 550
Melbourne, Florida 32901
Phone: 305 727-2020

WASH. D.C. OFFICE
9301 Annapolis Road
Lanham, Maryland 20801
Phone: 301 577-3001

Final Technical Report
for

DIGITAL TERRAIN DATA COMPACTION USING ARRAY ALGEBRA

Performed for:

USA-ETL
Fort Belvoir, Virginia

Under:

Contract No. DAAG53-76-C-0127

Performed by:

Urho Rauhala
Stephen Gerig

DBA Systems, Inc.
P.O. Drawer 550
Melbourne, Florida 32901

November 1976

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ETL-0108	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DIGITAL TERRAIN DATA COMPACTION USING ARRAY ALGEBRA		5. TYPE OF REPORT & PERIOD COVERED Contract Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Urho Rauhala, Stephen Gerig		8. CONTRACT OR GRANT NUMBER(s) DAAG53-76-C-0127
9. PERFORMING ORGANIZATION NAME AND ADDRESS DBA Systems, Inc. P.O. Drawer 550 Melbourne, Florida 32901		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060		12. REPORT DATE November 1976
		13. NUMBER OF PAGES 55
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

DDC

JUN 23 1977

B

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report investigates the applicability of array algebra digital terrain modeling and data compaction. Two options were evaluated for converting the collected data into regularly spaced terrain elevations. First, an array prediction is performed of the data directly in the epipolar coordinate frame. This approach allows for data compaction and subsequent evaluation at uniform intervals in a gridded map coordinate system. Second, next page

DD FORM 1473 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

405 059

20. continued

cont.

the principle of array algebra in a piecewise translocation algorithm is applied. In this approach the non-gridded epipolar coordinates are first converted to regularly spaced elevation data and then subsequently compacted using the methods of array prediction. In addition to analyzing the mathematical equations required for terrain compaction, the computational requirements were analyzed for both sequential and parallel processors.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1. INTRODUCTION

This report, submitted in fulfillment of Data Item A001, Contract No. DAAG53-76-C-0127, presents the results of a study performed by DBA Systems, Inc. investigating the applicability of array algebra digital terrain modeling and data compaction. For the purpose of this study the data were assumed to be collected along epipolar lines.

During the course of this study two options for converting the collected data into regularly spaced terrain elevations were evaluated. The first of these is to perform an "Array Prediction", or functional representation, of the data directly in the epipolar coordinate frame (Section 3). This approach allows for data compaction and subsequent evaluation at uniform intervals in a gridded "map" coordinate system. The second option considered during this study applies the principle of array algebra in a piecewise "translocation algorithm" (Section 4). In this method the non-gridded epipolar coordinates are first converted (translocated) to regularly spaced elevation data and then subsequently compacted using the methods of "array prediction".

During this study it was found that efficient direct array prediction algorithms could be derived and that the explicit formation of the regression coefficients could be avoided. Using these algorithms the array requirement for rectangularly spaced data was removed; thereby allowing the boundaries of the area to assume arbitrary shapes. In delineating these methods several examples of direct prediction coefficients, or "k-vectors", are derived. In typical cases, the knowledge of ten (10) such coefficients replaces the data handling as conventionally required for the explicit formulation and solutions of normal equations.

The algorithms developed for translocation demonstrate that between 2 and 10 scalar multiplications are required for each point manipulated. During the course of this effort the various alternatives for performing the operation were examined.

In addition to analyzing the mathematical equations required for terrain compaction, the computational requirements were analyzed for both sequential and parallel processors. Based upon this analysis it appears that the associative array processor is ideally suited for processing the derived algorithms. This is due to the feature of "sweeping while scanning", i.e. processing the algorithms first in one direction and immediately using the results in the other coordinate direction. It appears that only minimal storage is required to generate the one sweep characteristic for the algorithms, thereby avoiding multiple read-in (fetch) or read-out (store) cycles.

The remainder of this report describes, in detail the results of this effort. Section 2 provides some relevant background on the principle of array algebra and illustrates its application using simple functional examples. Sections 3 and 4 put forth the detailed mathematical developments and analysis of the algorithms for direct prediction and translocation. The final section (5) presents the conclusions and recommendations based upon the study.

2. BASIC PRINCIPLES OF ARRAY ALGEBRA

This section is devoted to an exposition of the basic principles of Array Algebra, centered around the notions of array multiplication and array least squares regression. The first subsection introduces Array Algebra in the simplest situation of two dimensions, and contains a numerical example to illustrate the ideas. The second subsection presents the theory for higher dimensions. The third subsection discusses the savings in computations that array algebra solutions to linear equations produce compared to conventional least squares techniques.

2.1 Array Algebra in Two Dimensions with a Numerical Example

Let m and n be positive integers. By a 2-dimensional array of size (n x m) is meant a collection of real (or complex) numbers, indexed by two indices $\imath$ and $\jmath$, which assume integer values between 1 and n and m, respectively. An individual number of the collection, called an element of the array, is denoted by the form $x_{\imath\jmath}$, and the array itself is abbreviated $[x_{\imath\jmath}]$.

As an ordinary matrix is also a collection of numbers indexed by two indices, the separate notion of 2-dimensional array may appear superfluous. It will be seen below, however, that the roles of arrays and matrices are different.

Matrices will be denoted by capitals from the beginning of the alphabet, arrays by capitals from the end.

Although a 2-dimensional array has the same structure as a matrix, its role in mathematics is in several ways like that of a vector. In particular, two arrays $X = [x_{\imath\jmath}]$ and $Y = [y_{\imath\jmath}]$ of the same size (n x m) may be added together to form a new (n x m) array as follows:

$$X + Y = [x_{\imath\jmath} + y_{\imath\jmath}] .$$

This operation is called array addition. Also, if c is a scalar, that is, a real (or complex) number, then c and the array X may be multiplied to form a new (n x m) array as follows:

$$cX = [cx_{ij}].$$

This operation is called array scalar multiplication.

The two array operations defined above satisfy the same rules as do the corresponding operations on vectors. Because of this, the set of all arrays of a given size, together with the two array operations, have the structure of a vector space. For given size (n x m), this space will be denoted $R_{n,m}$, and will be called (n x m) array space.

There is a third type of operation on an array space $R_{n,m}$ which resembles the operation of multiplying a vector by a matrix to produce a new vector.

Let $X = [x_{ij}]$ be an (n x m) array; let $A = [a_{hk}]$ be a (p x q) matrix. If q=n, an operation denoted o_1 on A and X that yields a new (p x m) array is defined as follows[1]:

$$\underset{(p,n)}{A} \; o_1 \; \underset{(n,m)}{X} = \underset{(p,m)}{\left[\sum_{k=1}^{n} a_{hk} x_{kj} \right]} = [y_{hj}].$$

The new array is indexed by h and j, in that order and equates to the hth row of A times the jth column of X.

If q = m, a second operation denoted o_2 on A and X that yields a new (n x p) array is defined as follows[2]:

$$\underset{(p,m)}{A} \; o_2 \; \underset{(n,m)}{X} = \underset{(n,p)}{\left[\sum_{k=1}^{m} a_{hk} x_{ik} \right]} = [y_{ih}].$$

The new array is indexed by i and h, in that order and equates to the hth row of A times the ith row of X.

---

1 The operation A o_1 is denoted by a superscript A^1 in Rauhala (1974)
2 Denoted A^2X in Rauhala (1974)

Both of the above operations will be called array multiplication. They have the property of being linear operations. That is,

$$A\ o_1\ (cX + dY) = cA\ o_1\ X + dA\ o_1\ Y,$$

and

$$A\ o_2\ (cX + dY) = cA\ o_2\ X + dA\ o_2\ Y.$$

These equalities follow directly from the definition of the operations, in the same way as do the corresponding operations on vectors.

Three further important equalities are as follows:

$$(AB)\ o_1\ X = A\ o_1\ (B\ o_1\ X),$$

$$(AB)\ o_2\ X = A\ o_2\ (B\ o_2\ X), \qquad \text{where } (AB) = \text{matrix product of A and B}$$

and

$$A\ o_1\ (B\ o_2\ X) = B\ o_2\ (A\ o_1\ X).$$

These equations are derived by observing that if $A = [a_{hk}]$, $B = [b_{\ell m}]$ and $X = [x_{ij}]$, then both sides of the first equation may be written as

$$[\sum_i \sum_k a_{hk}\ b_{ki}\ x_{ij}],$$

and both sides of the second equation may be written as

$$[\sum_k \sum_j a_{hk}\ b_{kj}\ x_{ij}].$$

For the third equation, observe that

$$\sum_i a_{hi} \sum_j b_{kj}\ x_{ij} = \sum_j b_{kj} \sum_i a_{hi}\ x_{ij}.$$

A fourth operation on arrays is the <u>dot</u> product, which is similar to the dot product for vectors. Namely, if $X = [x_{ij}]$ and $Y = [y_{ij}]$ are both arrays of size $(n \times m)$, then

$$X \cdot Y = \sum_i \sum_j x_{ij}\ y_{ij};$$

and is a number equal to the sum of the products of all corresponding elements in the two arrays [1].

---

[1] *Denoted RS(X*Y) in Rauhala (1974)*

The value of a dot product is a scalar. The dot product operation is only defined for arrays of the same size. If complex numbers are involved, the above definition must be changed to

$$X \cdot Y = \sum_i \sum_j x_{ij} \overline{y}_{ij},$$

where the bar denotes complex conjugation.

Recall that if $A = [a_{hk}]$ is an $(n \times m)$ matrix, then the transpose A^T of A is the $(m \times n)$ matrix whose (k,h)th entry a'_{kh} is a_{hk}. The transpose operation is related to the dot product for vectors as follows. If v is an m-vector, and u is an n-vector, then

$$(Av) \cdot u = v \cdot (A^T u).$$

There are corresponding equations for the array dot product and the array multiplication of matrices and arrays: if A is an $(n \times m)$ matrix, and X and Y are $(m \times p)$ and $(n \times p)$ arrays respectively, then

$$(A \circ_1 X) \cdot Y = X \cdot (A^T \circ_1 Y).$$

This equality follows from the fact that

$$\sum_i \sum_k \left(\sum_j a_{ij} x_{jk}\right) y_{ik} = \sum_j \sum_k x_{jk} \left(\sum_i a_{ij} y_{ik}\right).$$

If now X and Y are $(p \times m)$ and $(p \times n)$ arrays respectively, then

$$(A \circ_2 X) \cdot Y = X \cdot (A^T \circ_2 Y).$$

This follows from

$$\sum_i \sum_k \left(\sum_j a_{kj} x_{ij}\right) y_{ik} = \sum_i \sum_j x_{ij} \left(\sum_k a_{kj} y_{ik}\right).$$

In case complex numbers are involved, the transpose operation must be replaced by the complex transpose*; namely, A* is the matrix whose (j,i)th element a'_{ji} equals the complex conjugate $\overline{a}_{ij}$ of the (i,j)th element a_{ij} of A.

The numerical example of this subsection is a 2-dimensional array least squares regression.

Let p_1 and p_2 each be a finite set of parallel lines in the Euclidean plane E_2, but such that the lines of p_1 are not parallel to the lines of p_2. The set of all points that are intersections of lines p_1 and p_2 is called a bounded grid in E_2.

In this example, let p_1 consist of the lines $x = 0$, $x = 1$, $x = 4$, $x = 5$; p_2 consist of the lines $y = -4$, $y = 1$, $y = 2$. The associated bounded grid G therefore consists of the 12 pairs of points (x,y) such that x equals either 0, 1, 4 or 5, and y equals either -4, 1 or 2.

Associated with G is a pair (i, j) of indices, $1 \leq i \leq 4$, $1 \leq j \leq 3$, such that the (i, j)th element of G is the intersection of the ith line of p_1 and the jth line of p_2; the set of parallel lines is itself ordered either from left to right, or, if horizonal, from bottom to top.

To the (i, j)th point of G, let there correspond a number u_{ij} such that the (4x3) array $U = [u_{ij}]$ is as follows

$$U = \begin{bmatrix} 0 & 2 & -1 \\ 1 & 1 & 1 \\ -1 & 0 & 2 \\ -3 & -1 & 0 \end{bmatrix}.$$

The elements u_{ij} of U can be considered as values of an empirically given function at the grid points (x_i, y_j) of correspoinding index pairs (i, j). Analogously, if the pairs (x,y) of the grid are latitudes and longitudes of points on the Earth, the u_{ij} could represent radial elevations above a regular figure such as a sphere or spheroid fitted to the surface of the Earth.

The problem is to determine the coefficients w_{hk} in the math model

$$z = \sum_h \sum_k w_{hk} \; f_h(x) \, g_k(y)$$

which "best" fit the empirical values U. Two finite sets of functions $f_1, \ldots, f_r$ and $g_1, \ldots, g_s$ of a single real variable are given, such that the domains of the f_h include all of the first coordinates of the grid points in G, and the domains of the g_k include all of the second coordinates of the grid points of G. Then real values for variables w_{hk}, $1 \leqq h \leqq r$, $1 \leqq k \leqq s$ are to be determined, so that the sum

$$\sum_{i=1}^{4} \sum_{j=1}^{3} \left(\sum_{h=1}^{r} \sum_{k=1}^{s} f_h(x_i) g_k(y_j) \, w_{hk} - u_{ij} \right)^2$$

is a minimum.

In the present example, let $r=3$ and $s=2$. Also, assume $f_h(x) = x^{h-1}$, and $g_k(y) = y^{k-1}$. The functions f_h and g_k and the grid G give rise to two matrices $A = [f_h(x_i)]$ and $B = [g_k(y_j)]$ of sizes (4x3) and (3x2) respectively. Namely

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & -4 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

Form the (3x2) array $W = [w_{hk}]$. Then using the array dot product, the above sum of squares may be expressed as

$$(B \, o_2 \, A \, o_1 \, W - U) \cdot (B \, o_2 \, A \, o_1 \, W - U).$$

The objective is to express the solution value for the array W in this minimization as an array product involving the array U. This is called array least squares regression.

For any (nxm) matrix C of rank m, for which $m \leqq n$, set $C^{\ell} = (C^T C)^{-1} C^T$, or, if C contains complex numbers, $C^{\ell} = (C^* C)^{-1} C^*$. Then as shown below, the solution value for W is

$$W_0 = B^{\ell} \, o_2 \, A^{\ell} \, o_1 \, U.$$

Recall that two vectors x and y in a vector space are orthogonal if their dot product $x \cdot y$ equals 0. Using the analogous notion of orthogonality in the array spaces, W_0 may be characterized by the condition that $B\ o_2\ A\ o_1\ W_0 - U$ is orthogonal to $B\ o_2\ A\ o_1\ W$ for all arrays W in $R_{r,s} = R_{3,2}$. Geometrically this means that the array $B\ o_2\ A\ o_1\ W_0 - U$ is orthogonal to the subspace of all $B\ o_2\ A\ o_1\ W$, and so that $B\ o_2\ A\ o_1\ W_0$ is the element of this subspace closest to U. It must therefore be shown that

$$(B\ o_2\ A\ o_1\ W) \cdot (A\ o_1\ B\ o_2\ B^{\ell}\ o_2\ A^{\ell}\ o_1\ U - U) = 0$$

for all W in $R_{3,2}$. This equation may be rewritten as

$$W \cdot (B^T\ o_2\ A^T\ o_1\ A\ o_1\ B\ o_2\ B^{\ell}\ o_2\ A^{\ell}\ o_1\ U - B^T\ o_2\ A^T\ o_1\ U) = 0.$$

But

$$\begin{aligned} &B^T\ o_2\ A^T\ o_1\ A\ o_1\ B\ o_2\ B^{\ell}\ o_2\ A^{\ell}\ o_1\ U \\ &= (B^T B B^{\ell})\ o_2\ (A^T A A^{\ell})\ o_1\ U \\ &= B^T\ o_2\ A^T\ o_1\ U. \end{aligned}$$

Thus the right-hand side of the above dot product equals the zero array, so the dot product itself equals 0 for all W in $R_{3,2}$. Hence $W_0 = B^{\ell}\ o_2\ A_0^{\ell}\ o_1\ U$. In order to compute the value W_0 of W for the example, it is therefore necessary for form A^{ℓ}, B^{ℓ}, and to evaluate $B^{\ell}\ o_2\ A^{\ell}\ o_1\ U$. The (3x4) matrix A^{ℓ} is as follows:

$$\begin{bmatrix} .86765 & .22059 & -.22059 & .13235 \\ -.77206 & .53676 & .71324 & -.47794 \\ .12500 & -.12500 & -.12500 & .12500 \end{bmatrix}.$$

The (2×3) matrix B^{ℓ} equals

$$\begin{bmatrix} .27417 & .35484 & .37097 \\ -.17742 & .06452 & .11290 \end{bmatrix}.$$

Using these values for A^{ℓ} and B^{ℓ}, the array $W_0 = B^{\ell} \circ_2 A^{\ell} \circ_1 U$ is found to be

$$\begin{bmatrix} .25546 & -.01305 \\ 1.17161 & .05159 \\ -.28831 & .01008 \end{bmatrix}.$$

2.2 Array Algebra in Higher Dimensions

To simplify the writing of arrays in higher dimensions, the following notion of index vector is introduced. Let $\jmath_1 \ldots \jmath_N$ be a sequence of integer valued indices such that $\jmath_k$ takes values 1 to n_k. Form the index vector $J = (\jmath_1, \ldots, \jmath_N)$. Then J takes as values the vectors with integer coordinates in the hyperrectangle Q which consists of all vectors $(x_1, \ldots, x_N)$ such that $1 \leq x_k \leq n_k$, $1 \leq k \leq N$.

An N-dimensional array of size $(n_1 \times n_2 \times \ldots \times n_N)$ is a collection of real (or complex) numbers, indexed by N indices $j_1, \ldots, j_N$, or more compactly, by the index vector $J = (j_1, \ldots, j_N)$. The index vector J takes as values vectors with integer coordinates in a hyperrectangle Q. An array element is written in the form x_J, and the array itself is abbreviated $[x_J]$.

As in the case of two-dimensions, the set $R_{n_1, \ldots, n_N}$ of all arrays of size $(n_1 \times \ldots \times n_N)$ form a vector space, called an array space. The two operations of array addition and scalar multiplication are defined by

$$[x_J] + [y_J] = [x_J + y_J],$$

and

$$c[x_J] = [cx_J].$$

A matrix $A = [a_{ij}]$ and an N-dimensional array $[x_J]$ can be multiplied together in N different ways, each called an array multiplication[1]. Namely, if A has size $(m \times n)$ and $[x_J]$ has size $(n_1 \times \ldots \times n_N)$, then if $n = n_k$ let[2]

$$A \circ_k X = (\sum_{j_k} a_{i j_k} x_J).$$

$A \circ_k X$ has size $(n_1 \times \ldots \times n_{k-1} \times m \times n_{k+1} \times \ldots \times n_N)$, and index vector $(j_1, \ldots, j_{k-1}, i, j_{k+1}, \ldots, j_N)$.

As for the case of two dimensions, array multiplication in N dimensions has the following properties:

$$A \circ_k (cX + dY) = cA \circ_k X + dA \circ_k Y,$$

$$(AB) \circ_k X = A \circ_k B \circ_k X,$$

$$A \circ_k B \circ_h X = B \circ_h A \circ_k X,$$

for $k \neq h$. For $k = h$, the second relation shows that the third relation holds only for A and B such that $AB = BA$.

---

[1] *Termed R matrix multiplications in Rauhala (1974)*
[2] *Denoted as* $A^k X$ *in Rauhala (1974)*

The dot product of two arrays X and Y of the same size is defined by

$$X \cdot Y = \sum_{J} x_J \, y_J ,$$

where J varies over the vectors with integer coordinates in the hyperrectangle Q.

To formulate the problem of N-dimensional array least squares regression, the concept of a bounded grid in N-dimensional Euclidean space E_N must first be defined.

Let n_k be a positive integer for $1 \leqq k \leqq N$. For each k, let $x_1^{(k)}, \ldots, x_{n_k}^{(k)}$ be real numbers. The set G of all vectors in E_N of the form $x_J = (x_{j_1}^{(1)}, \ldots, x_{j_N}^{(N)})^T$, where $1 \leqq j_k \leqq m_k$, $1 \leqq k \leqq N$, is called a rectangular bounded grid in E_N.

Let B be a nonsingular matrix of real numbers of size (NxN). Then the set $D = BG$ of all vectors Bx_J, where x_J is a vector in the rectangular bounded grid G, is called a bounded grid in E_N.

Let $e_1, \ldots, e_N$ be the standard orthonormal basis for E_N, so $e_1 = (1,0,\ldots,0), \ldots,$ $e_N = (0,\ldots,0,1)$. Then $Be_1, \ldots, Be_N$ is also a basis in E_N, but it is in general not rectangular. Set $b_k = Be_k$. If $y = (y_1, \ldots, y_N) = \Sigma\, y_k e_k$, then $B_y = B(\Sigma\, y_k e_k) =$ $\Sigma\, y_k\, Be_k = \Sigma\, y_k\, b_k$. Therefore the elements of the bounded grid D may be expressed as $\sum_k x_{j_k}^{(k)} b_k$.

Let $U = [u_J]$ be an N-dimensional array of size $(n_1 \times \ldots \times n_N)$ whose elements are in practice values of an empirically defined function at the points B_J of the bounded grid D in E_N.

For each k, $1 \leqq k \leqq N$, let m_k be a positive integer, and let $f_{k i_k}$, $1 \leqq i_k \leqq m_k$, be functions of a single real variable, such that for each index vector J, the k th coordinate coefficient $x_{j_k}^{(k)}$ of Bx_J with respect to the basis $b_1, \ldots, b_N$ belongs to the domains of $f_{k1}, \ldots, f_{k m_k}$, $1 \leqq k \leqq N$. Let A_k be the $(n_k \times m_k)$ matrix $[f_{k i_k}(x_{j_k}^{(k)})]$.

The problem of array least squares regression is to find the array W of size $(m_1 \times \ldots \times m_N)$ such that the dot product

$$(A_1 \circ_1 A_2 \circ_2 \ldots A_N \circ_N W - U) \; (A_1 \circ_1 A_2 \circ_2 \ldots A_N \circ_N W - U)$$

is a minimum.

By an argument, based on orthogonality, entirely similar to the case of two dimensions, the solution value of W is

$$W_0 = A_1^{\ell} \circ_1 A_2^{\ell} \circ_2 \ldots A_N^{\ell} \circ_N U,$$

where

$$A_k^{\ell} = (A_k^T A_k)^{-1} A_k^T .$$

In case the elements of U are measured quantities, and the variances of all the measurements are <u>the same</u> and equal to the constant σ^2, then the maximum likelihood *estimate* of U of the form $A_1 \circ_1 A_2 \circ_2 \ldots A_N \circ_N W$ is the value of this expression for which W equals the W_0 above. This follows readily from the fact that the covariance matrix of the observations is $\sigma^2 I$, where I is the identity matrix of order $\prod_{k=1}^{N} n_k$.

2.3 Computational Advantages of Array Algebra and Array Regression, and Suitability for Programming on a Parallel Array Processor

The role of array algebra in computational applications consists of being a formalism for an extremely efficient method of solving systems of multilinear equations in which certain of the sets of quantities involved are associated with a bounded grid of points in some higher dimensional Euclidean space. Both the method of solution and the formalism of array algebra arise out of this association.

By the use of the operation of array multiplication, the amount of computation required for the solution of such a multilinear system, e.g., in the form $A_1^{-1} o_1 \ldots A_N^{-1} o_N U$, where all the matrices $A_1, \ldots, A_N$ have the same size $(n \times n)$, is, when done by sequential processing, proportional to the first power of the number n^N of parameters, and not to the third power $(n^N)^3$ as in the conventional linear solution.

The array solution method may also be compared with another fast algorithm for solving systems of linear equations of a special type, namely the Fast Fourier Transform. For this algorithm, the amount of computation required is proportional to the product of the number of parameters times the logarithm of the number of parameters. Moreover, this level of efficiency is only attainable when the number of parameters is a power of 2.

If A_k is an $(m_k \times n_k)$ matrix, and X is an $n_1 \times \ldots \times n_N$ array, then on a parallel array processor, the operation $A \, o_k \, X$ requires $m_k n_k$ parallel multiplications. Thus array products of the form $A_1 \, o_1 \ldots A_N \, o_N X$ or $A_1^{\ell} \, o_1 \ldots A_N^{\ell} \, o_N U$, in which the array U has size $m_1 \times \ldots \times m_N$, require a sequence of N subsequences of $m_1 n_1, \ldots, m_N n_N$ parallel multiplications respectively. This shows that array algebra operations and array regression are eminently suited for implementation on a parallel array processor.

3. APPLICATION OF ARRAY LEAST SQUARES REGRESSION TO DIGITAL TERRAIN DATA COMPACTION IN THE EPIPOLAR COORDINATE SYSTEM

The basis for the application of array least squares regression to the compaction of digital terrain data in the epipolar coordinate system is that the elevation observations are made at a set of points which in the epipolar coordinate plane form a grid. This implies that, in the terminology of Section 2, the observations form a 2-dimensional array. The functions that are used in the regression have a special form which is appropriate for terrain modelling; these functions are described in subsection 3.1. The structure of the compaction, the coefficients for the regression, and the array algebra formula that expresses these coefficients in terms of the observations are discussed in subsection 3.2. Subsection 3.3 addresses the problem of specifying the values of certain parameters that determine the individual functions of the regression. These parameters are not solved for, but are specified in advance on the basis of the general structure of the terrain being modelled and the density of the elevation observations made by the epipolar scanner. The recovery from the compaction of the elevations at all of the observation points, for input to the contour plotter, is described in subsection 3.4.

3.1 Functions Occurring in the Array Least Squares Regression for Digital Terrain Data Compaction

Let $U = [u_{ij}]$ be the 2-dimensional array of elevations observed by an epipolar scanner at a set of observation points which in the epipolar coordinate system form a bounded grid D. By Section 2.2, there is a nonsingular linear transform B and a *rectangular bounded grid* G such that $D = BG$. If $b_1 = B(1,0)^T$ and $b_2 = B(0,1)^T$, then for each (x,y) in the bounded grid G, $xb_1 + yb_2$ is an element of the bounded grid D. Let $x_1, \ldots, x_N$ and $y_1, \ldots, y_M$ be real numbers such that G consists of the vectors (x_i, y_j); then D consists of all $x_i b_1 + y_j b_2$, for $1 \leq i \leq N$ and $1 \leq j \leq M$.

According to Section 2, the problem of array least squares regression in two dimensions, as applied to the array U of observations at grid points in D, is to determine the coefficients a_{hk} in a linear combination

$$\sum_h \sum_k a_{hk} \; f_h(x) \, g_k(y)$$

such that the sum of squares

$$\sum_i \sum_j \left(\sum_h \sum_k a_{hk} \; f_h(x_i) \, g_k(y_j) - u_{ij} \right)^2$$

is a minimum.

The form of the functions f_h and g_k to be considered for the present application to digital terrain data compaction is as follows. Let c be a real number, $c < 0$. Then the function $\exp(cx^2)$ is called a <u>Gaussian error function</u>, and the functions f_h and g_k will be translates of functions of this form. That is, there are constants c_h and s_h, d_k and t_k such that

$$f_h(x) = \exp(c_h (x - s_h)^2),$$

and

$$g_k(y) = \exp(d_k (y - t_k)^2).$$

If $1 \leq h \leq H$ and $1 \leq k \leq K$, then the points (s_h, t_k) are called <u>node points</u> for the regression.

Gaussian error functions take the value 1 at the origin 0, and are symmetric about the origin because the argument in the exponential is squared. They also decrease rapidly to 0 as the argument grows in magnitude, the rapidity being greater the more negative the constant c.

The motivation for the choice of translates of Gaussian error functions for the regression is that for the prediction or estimation of the elevation at a point with coordinates $x_0 b_1 + y_0 b_2$, by the evaluation of the functional model

$$F(x,y) = \sum_h \sum_k a_{hk} \; f_h(x) \, g_k(y)$$

at (x_0, y_0), the values $f_h(x_0)$ and $g_k(y_0)$ for node points (s_h, t_k) not very near to (x_0, y_0) are negligibly small, so that $F(x_0, y_0)$ is adequately expressed by a sum

$$\sum_h \sum_k a_{hk} \; f_h(x_0) \, g_k(y_0),$$

where h and k take only a small range of values corresponding to node points (s_t, t_k) very close to (x_0, y_0).

A more detailed discussion of the origin and suitability of the above choice of translates of Gaussian error functions for the regression is contained in Kraus and Mikhail [2].

3.2 The Nature of the Compaction and the Array Algebra Formula for the Coefficients

The functional model

$$F(x,y) = \sum_h \sum_k a_{hk} \; f_h(x) \, g_k(y)$$

being the one used to estimate terrain elevations, it follows that the compaction of the terrain data consists of the following:

(a) The set of node points (s_h, t_k), and the coefficients c_k and d_k, $1 \leq h \leq H$, $1 \leq k \leq K$. These are selected on a basis to be discussed below.

(b) The set of coefficients a_{hk}, $1 \leq h \leq H$, $1 \leq k \leq K$. These are computed using the expression given in Section 2 for the solution to the 2-dimensional array least squares regression problem.

Relative to the computation of the coefficients a_{hk}, it will happen in practice that the number NM of observation points is very large, as may be the number HK of coefficients a_{hk}. Nevertheless in the array algebra determination of the coefficients a_{hk}, the only matrices that need to be inverted are the (HxH) matrix $A_1^T A_1$ and the (KxK) matrix $A_2^T A_2$, where

$$A_1 = [f_h(x_i)], \qquad \text{of size } (N\times H),$$

and

$$A_2 = [g_k(y_j)], \qquad \text{of size } (M\times K).$$

Setting $A_1^{\ell} = (A_1^T A_1)^{-1} A_1^T$ and $A_2^{\ell} = (A_2^T A_2)^{-1} A_2^T$, then in the notation of Section 2, the array $[a_{hk}]$ is equal to

$$A_1^{\ell} \circ_1 A_2^{\ell} \circ_2 U.$$

This formula is, by the discussion at the end of Section 2.2, valid for measured quantities for which all the associated variances are taken to have the same values.

A slightly more general situation may be accommodated by the array algebra; this is the case in which the covariance matrix for all the observations is expressible as a <u>tensor product</u> $C_1 \otimes C_2$. Here $C_1 = [c'_{ij}]$ is of size (NxN), $C_2 = [c''_{ij}]$ is of size (MxM), and $C_1 \otimes C_2$ is by definition equal to the (NM x NM) matrix $[C_1\, c''_{ij}]$. In this case the array of coefficients $[a_{hk}]$ equals

$$((A_1^T C_1^{-1} A_1)^{-1} A_1^T C_1^{-1}) \circ_1 ((A_2^T C_2^{-1} A_2)^{-1} A_2^T C_2^{-1}) \circ_2 U.$$

In practice, the matrices C_1 and C_2 will be diagonal, so their inversions are performed by inverting their scalar diagonal elements term-by-term.

Also in practice, an empirically determined covariance matrix may be approximated more or less well by a positive definite tensor product matrix. This could be used in the array formula above for the regression coefficients to produce values for these which are better in the sense that their variances are smaller than would otherwise be if the covariance matrix of the observations was tacitly assumed to be a positive scalar multiple of the identity matrix, the scalar being realistically large.

3.3 The Specification of the Parameters in the Gaussian Error Functions

The functional model of the terrain elevations is a linear combination of products of pairs of functions f_h and g_k, each such function being selected as a translate of a Gaussian error function, hence of the mappings

$$x \to \exp(c(x-s)^2), \qquad y \to \exp(d(y-t)^2),$$

where $c, d < 0$ and s, t are parameters. This subsection addresses the problem of specifying the values of the parameters c and d, and s and t in all of the functions f_h and g_k occurring in the functional model.

Two considerations influence the choice of the parameters. These are computational feasibility and efficiency, and the amount of undulation of the terrain being modelled, i.e., whether it is level to broadly rolling, or very rolling to broken. These two considerations are interconnected, and will be discussed simultaneously.

To avoid ill-conditioning in the matrices to be inverted in the determination of the coefficients a_{hk}, it is necessary that for a given spacing of the node points the magnitudes of the parameters c_h and d_k in f_h and g_k respectively are not too large or too small. If the coordinate system is so chosen that the mean distance between adjacent node points is 1, then the c_h and d_k should be not more than $\ell n(0.8) \doteq -.223$, and not less than $\ell n(0.2) \doteq -1.609$. Within these limits, all the c_h are assigned a single value c,

and all the d_k are assigned a single, possibly different, single value d. These assignments are made with a view to simplifying computations. The actual values of c and d, within the limits set above, will be farther from 0 for less rolling terrain, and closer to 0 for more rolling terrain.

As will be shown in 3.4, computational advantages occur if all the node points are also observation points. Then the node points will form a subgrid of the grid of observation points. This is especially true if the node point grid is obtained from the observation point grid by deleting every other row and every other column. This is the situation that will be considered here. The compaction will then consist of the at most $(N+1)(M+1)/4$ coefficients a_{hk}, the values of the parameters c_h, d_k, s_h, t_k being specifiable by only a few words.

It is clear that because the spacings of the observation points and the node points are closely related, and because the spacing of the node points and the values of the parameters c_h and d_k are also related, the observation point spacing must be chosen in a particular way. Namely, the spacing of the observation grid points is such that relative to the resulting subgrid of node points, values of c_h and d_k can be chosen in the interval from -1.609 to -.223, as described above, so that undulations of the terrain are adequately modelled by the function

$$F(x,y) = \sum_h \sum_k a_{hk} \ f_h(x) \ g_k(y).$$

3.4 Recovery of Observation Point Values from the Compaction

In the most general situation, in which the node points do not necessarily form a subgrid of the grid of observation points, elevation values at grid points are recovered from the compaction in the form of the estimates obtained by evaluating the functional model at the observation grid points. If, as in subsection 3.2,

$A_1 = [f_h(x_i)]$ and $A_2 = [g_k(y_j)]$, where (x_i, y_j), $1 \leq i \leq N$, $1 \leq j \leq M$, are the observation grid points, then

$$A_1 \circ_1 A_2 \circ_2 [a_{hk}]$$

is the array of observation point estimated elevations.

Because of the choice of the functions f_h and g_k, the matrices A_1 and A_2 may in practice be replaced by banded matrices A_1' and A_2', obtained from A_1 and A_2 by replacing all of the negligibly small elements in them by 0. In a single column, these will be all elements sufficiently far from the diagonal; i.e., corresponding to numbers x_i or y_j which are far enough away from the s_h or t_k for that column respectively, so that $f_h(x_i) = \exp(c_h(x_i - s_h)^2)$ or $g_k(y_j) = \exp(d_k(y_j - t_k)^2)$ are negligibly small.

In practice, for this general situation, the array of elevations at the grid of observation points is estimated by

$$A_1' \circ_1 A_2' \circ_2 [a_{hk}] .$$

If the bandwidth of A_1' is $2b_1$ and that of A_2' is $2b_2$, then $(2b_1+1)NK + (2b_2+1)NM$ scalar multiplications are required in evaluation of the NM elevation grid.

When the node points form a subgrid of the grid of observation points, as described in subsection 3.3, then by a transformation of the functional model, the array of coefficients $[a_{hk}]$ may be replaced by a new coefficient array $[F(s_p, t_q)]$ of values of the functional model at the grid of node points (s_p, t_q), $1 \leq p \leq H$, $1 \leq q \leq K$. This does not achieve any further compaction, but does reduce the amount of computation necessary to recover the elevations at all the observation grid points.

Let $B_1 = [f_h(s_p)]$ and $B_2 = [g_k(t_q)]$, so that B_1 and B_2 have sizes $(H \times H)$ and $(K \times K)$ respectively, and are moreover nonsingular. Then

$$[F(s_p, t_q)] = B_1 \circ_1 B_2 \circ_2 [a_{hk}];$$

hence

$$[a_{hk}] = B_1^{-1} \circ_1 B_2^{-1} \circ_2 [F(s_p, t_q)].$$

Therefore the estimated elevations at the observation grid points are expressible in terms of the elevation estimates $F(s_p, t_q)$ at the node points by

$$[F(x_i, y_j)] = A_1 \circ_1 A_2 \circ_2 (B_1^{-1} \circ_1 B_2^{-1} \circ_2 [F(s_p, t_q)])$$

$$= (A_1 B_1^{-1}) \circ_1 (A_2 B_2^{-1}) \circ_2 [F(s_p, t_q)].$$

Moreover the array $[F(s_p, t_q)]$ is expressible in terms of the array U of observed elevations by the equations

$$[F(s_p, t_q)] \approx B_1 \circ_1 B_2 \circ_2 [a_{hk}]$$

$$\approx B_1 \circ_1 B_2 \circ_2 A_1^{\ell} \circ_1 A_2^{\ell} \circ_2 U$$

$$\approx (B_1 A_1^{\ell}) \circ_1 (B_2 A_2^{\ell}) \circ_2 U$$

$$\approx (A_1 B_1^{-1})^{\ell} \circ_1 (A_2 B_2^{-1})^{\ell} \circ_2 U.$$

This shows that the array $[F(s_p, t_q)]$ may be used as an array of coefficients of a functional model for the terrain, and that this model is obtained from the original functional model by means of simple array operations. The new functional model and the computational techniques described below are together called <u>direct prediction</u>.

As explained in subsection 3.3, the grid of node points was selected from the grid of observation points by deleting every other row and every other column of observation points. Let the epipolar coordinate system be normalized so that the node points have consecutive integer coordinates. This requires that the observation points be equally spaced and form a rectangular grid in the epipolar plane. Then the observation points

themselves include the node points plus sets of points of the form $(n+\frac{1}{2}, m)$, $(n, m+\frac{1}{2})$ and $(n+\frac{1}{2}, m+\frac{1}{2})$, where m and n are integers. Once the values of N, M, H, K, and the common values c and d of the c_h and d_k respectively are assigned, the matrices A_1, A_2, B_1^{-1} and B_2^{-1} may be computed once and for all.

Since the values of F at the node points are already explicitly known, being the coefficients of the new functional model, it is only necessary, for the recovery of the elevations at the observation grid points, to do computations for the points in the observation grid of the form $(n, m+\frac{1}{2})$, $(n+\frac{1}{2}, m)$, $(n+\frac{1}{2}, m+\frac{1}{2})$. Accordingly let E_1 and E_2 be the matrices $[f_h(n+\frac{1}{2})]\, B_1^{-1}$ and $[g_k(m+\frac{1}{2})]\, B_2^{-1}$ respectively.

The matrices E_1 and E_2 have the following important property: they may be approximated very closely by matrices E_1' and E_2' which are <u>banded</u> and <u>circulant</u>, as defined below, except for the first and last few rows. These properties are inherited by E_1 and E_2 from the matrices

$$[f_h(n+\tfrac{1}{2})] = [\exp(c(n+\tfrac{1}{2}-h)^2)],$$

$$[g_k(m+\tfrac{1}{2})] = [\exp(d(m+\tfrac{1}{2}-k)^2)],$$

$$B_1 = [\exp(c(n-h)^2)],$$

and

$$B_2 = [\exp(d(m-k)^2)],$$

where s_h *and* t_k *have been set equal to h and k respectively. To say that a matrix is circulant is to say that each row, except for the top, is obtained from the row above it by a one column right translation.* Because their elements are functions of the difference of the row and column indices, the matrices listed above have this property.

Because of their special structure, E_1' and E_2' *are each essentially determined by a single row, and the only nonzero elements of the row are the ones within the band. Finally, symmetry properties of the matrices result in a row having the form*

$$0\ldots0\ e_b\ e_{b-1}\ \ldots\ e_2\ e_1\ e_1\ e_2\ \ldots\ e_{b-1}\ e_b\ 0\ldots0,$$

where 2b is the bandwidth. These rows are called k-vectors. In a matrix or array multiplication involving a matrix E_1' or E_2', a k-vector and a column would combine into an expression

$$e_1(z_{i,j} + z_{i+1,j}) + e_2(z_{i-1,j} + z_{i+2,j}) + \ldots$$
$$\ldots + e_b(z_{i-b+1,j} + z_{i+b,j}),$$

requiring b scalar multiplications.

The sequence $|e_1|, |e_2|, \ldots, |e_b|$ is a monotonically decreasing sequence of positive numbers. The value of b is chosen in the following manner. Let J be the flying altitude, R the base-to-height ratio, f the focal length of the camera. Also let the maximum height difference from the mean value of 2b consecutive observation points be less than p% of the flying altitude J. Finally, let q be the mean error of image coordinates. Then b must be such that

$$|e_{b+1}|\,(p/100)\,J < (J/f)\,R^{-1}\,q.$$

Typical values for these quantities are f = 15 cm., R = .625 (with image area equal to 23 x 23 sq. cm.), p = 5%, q = 10 μm. The quantity J cancels in the above inequality, which then becomes $|e_{b+1}| < 0.02$. To be on the safe side and to allow for round-off errors, 0.02 should perhaps be replaced by 0.005.

The following table expresses the values of the k-vector sequence $e_1, \ldots, e_b$ as a function of the values of $\exp(cx^2)$ at x = 1. Computations were performed by means of a banded array Cholesky algorithm.

TABLE 1

exp(c)	b	$e_1, \ldots, e_b$
.2	5	.5728, -.0924, .0183, -.0037, .0007
.4	7	.6149, -.1577, .0596, -.0237, .0094, -.0038, .0015
.6	11	.6298, -.1928, .0983, -.0560, .0330, -.0197, .0118, -.0071, .0042, -.0025, .0015
.8	20	.6353, -.2083, .1209, -.0823, .0601, -.0455, .0352, -.0276, .0218, -.0173, .0137, -.0109, .0087, -.0069, .0054, -.0043, .0034, -.0026, .0020, -.0015

The case $\exp(c) = .5 \approx 2^{-1}$ has a special advantage for computations.

TABLE 2

exp(c)	b	$e_1, \ldots, e_b$
.5	10	.624052, -.178300, .080523, -.039310, .019540, -.009756, .004876, -.002438, .001219, -.000609

After the first few terms, as for the other values of exp(c), the sequence is very nearly a geometric progression, with factor exp(c). This means that in a matrix or array multiplication containing a matrix E_1' or E_2' with the above k-vector, the k-vector and a column would combine into an expression, when $\exp(c) = 2^{-1}$,

$$
\begin{aligned}
&.624(z_i + z_{i+1}) - .1783(z_{i-1} + z_{i+2}) \\
&+.0805(z_{i-2} + z_{i+3}) + .004876\,[z_{i-6} \\
&+z_{i+7} - 2[z_{i-5} + z_{i+6} + 2[-(z_{i-4} + z_{i+5}) \\
&+2(z_{i-3} + z_{i+4})]] - 2^{-1}\,[z_{i-7} + \\
&z_{i+8} + 2^{-1}\,[-(z_{i-8} + z_{i+9}) + 2^{-1}(z_{i-9} + z_{i+10})]]].
\end{aligned}
$$

Here the second index j of z has been supressed to simplify the appearance of the formula.

The k-vector term of smallest magnitude included in the above sum is

$e_{10} = -.000609.$

The evaluation of the formula requires only four scalar multiplications. If the term $e_3 = .0805$ is included in the approximate geometric progression, only three scalar multiplications are needed.

In the densification, i.e., the recovery of the elevations at the observation grid points from node point values (n,m), the total number of scalar multiplications required for the observation grid points of the form $(n+\frac{1}{2},m)$ and $(n,m+\frac{1}{2})$ are 3(HK) + 3(HK) = 6HK. Using one set of the elevations so computed, the elevations at the observation grid points of the form $(n+\frac{1}{2},m+\frac{1}{2})$ are computed with 3HK additional scalar multiplications. The densification of HK node point values by 3HK elevations requires 9HK scalar multiplications or 3 multiplications per point. Equally rigorous conventional computational methods for this densification require on the order of $(HK)^3$ scalar multiplications. The accuracy required for the operation needs in practice to be to only three or four decimal places.

Let b_1 be the bandwidth in the x-direction, b_2 in the y-direction. The one-sweep principle of saving extra read-in-read-out operations requires $(2b_1+3)K$ words of core space for saving simultaneously $2b_1$ grid lines of node point values and 3K values of predictions $F(n+\frac{1}{2},m)$, $F(n,m+\frac{1}{2})$ and $F(n+\frac{1}{2},m+\frac{1}{2})$. Then read-in one row (or column if the sweep direction is changed) of the grid elevations only once, and the intermediate interpolations need not be output and again input if one proceeds as follows.

1. Read in the elevations of the $(n+b_1)$th grid row

2. Compute

   $F(n+\frac{1}{2},m)$

3. Using the resulting values $F(n+\frac{1}{2},m)$ compute

   $F(n+\frac{1}{2}, m+\frac{1}{2})$, $m = b_2, b_2+1, \ldots, K-b_2$

4. To get the values $F(n,m+\frac{1}{2})$, we use any of the $2b_1$ rows of elevations preferably a

   $F(n+b_1, m+\frac{1}{2})$

5. Now the following rows are no longer needed in the computations and may be output:

   The $(n-b_1)$th row of node point values

   $F(n+\frac{1}{2}, m+\frac{1}{2})$

   $F(n+b_1, m+\frac{1}{2})$

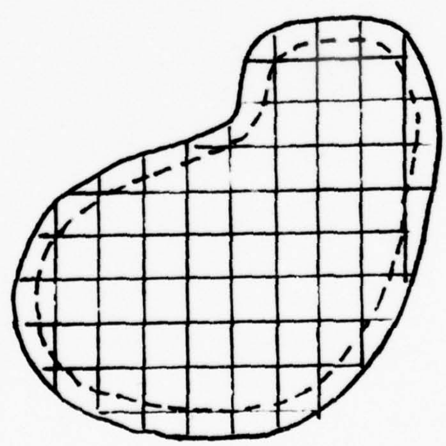
   $F(n+\frac{1}{2}, m)$

6. Replace n by n+1. Repeat procedure until $n=H$.

Notice that the last interpolated row of the $F(n+\frac{1}{2},m)$ and the $F(n+\frac{1}{2},m+\frac{1}{2})$ is for $n=N-b_1$. The last interpolated row of the $F(n,m+\frac{1}{2})$ is for $n=N-b_1$.

Because the node point elevations and the interpolated values only have 5 decimals (for computations only 3 or 4 decimals are needed if their local mean value is considered as a trend function) the stored $2b_1$ K words can be compressed to fewer but longer words on the general purpose computers, Gennery [1]. If extra core space were available, in addition to the necessary $2b_1$ K words, one could apply the above algorithm to batches of, say, p rows. Then the number of read-out, read-in operations during the one-sweep processing is decreased p-fold. The array processor is an ideal development in this respect because now the p rows can be processed in parallel as if only one row existed. Especially the more efficient read-in - read-out operations of the parallel processor speeds the processing compared with the sequential processor. Some closer comparisons with actual processor times of the two different methods was not performed during this study.

The principle of direct prediction in the two-dimensional case requires $2b_1+2b_2$ node point values $F(n,m)$ only in the closest area of $F(n+\frac{1}{2}, m+\frac{1}{2})$, $F(n, m+\frac{1}{2})$, and $F(n+\frac{1}{2}, m)$. This fact allows the boundaries of the area to assume arbitrary shapes such as



and the direct predictions still can be utilized inside the narrow zone of boundaries.

4. ARRAY TRANSLOCATION IN MAP COORDINATES TO PREPARE FOR ARRAY ALGEBRA DATA COMPACTION

In the map plane versus the epipolar plane, the points at which elevations are observed by the epipolar scanner do not form a grid. In the present section, a method called array translocation is described for determining elevations at a set of points called the elevation grid. The elevation grid is a bounded grid in the map plane, the elevations at whose points can be input to a contour plotter. The set of elevations at the points of the elevation grid may be compacted by the array algebra method of Section 3, the elevations being treated for this purpose as an array of observations at the points of a bounded grid. Subsection 4.1 describes the principle of array translocation. Subsection 4.2 contains examples and algorithms for different forms of array translocation.

4.1 The Principle of Array Translocation

In the present discussion, a set of points is specified which in the map coordinate system form a bounded grid. The map coordinate system is taken to be normalized so that thegrid is rectangular and the grid points have integer coordinates, consecutive grid points on a grid line having consecutive integer coordinates. This bounded grid is called the elevation grid. The problem is to compute estimates of elevations at the points of the elevation grid from a set of elevations observed by the epipolar scanner at points which do not form a grid in the map coordinate system.

The transformation from the epipolar plane to the map plane is assumed to be such that locally the departure from linearity is not too severe, and distances are nearly preserved. Because the observation points of the epipolar scanner form a grid in the epipolar plane, taken here to be rectangular, these observation points have locally a nearly gridded arrangement in the map plane.

The method of estimating elevations at the points of the elevation grid consists in locally interpolating or fitting low degree polynomials through the elevations at the observation points, and then evaluating these polynomials at the points of the elevation grid; a given polynomial is evaluated at those points of the elevation grid which are within the local region over which that polynomial has been fitted.

Any estimation technique of the above sort is called translocation. Array translocation consists of performing the estimations in two steps. The first step is an estimation of elevations at a set of intermediate points, with integer X-coordinates, by translocation involving polynomials of the X-variable only. The second step is an estimation of the elevations at the points of the elevation grid, from the elevations at the intermediate points, by translocation involving polynomials of the Y-variable only.

The first step of the array translocation is performed as follows. A positive integer N is specified in the interval from 1 to 5. The number N-1 will be the degree of the polynomials in the one variable X used for the local elevation estimate calculations. A second positive integer M is specified to be the number of consecutive points in an epipolar scan line to which a polynomial of degree N-1 will be fitted. Therefore in practice, $N \leqq M$. It is assumed for this discussion that the epipolar scanner scans along lines parallel to the X-axis in the epipolar plane, and hence in the map plane along curves roughly parallel to the map coordinate system X-axis.

Let $P_k = (x_k, y_k)$, $1 \leqq k \leqq M$, be consecutive points on an epipolar scan lines, with the displayed coordinates being relative to the map coordinate system. Let z_k, $1 \leqq k \leqq M$, be the elevation at P_k. Two polynomials F and G of degree N-1 are required. The polynomial F is fitted to the Y-coordinates y_k at the X-coordinates x_k. The polynomial G is fitted to the elevations z_k at the X-coordinates x_k. The differences $x_{k+1} - x_k$, $1 \leqq k \leqq M-1$, are by assumption all nearly equal to 1. For the purpose of determining the coefficients of F and G, these differences will be taken to be equal to 1.

It is known from the adjustment calculus that this approximation will not introduce any serious errors in the estimation. Let H be the largest integer less than or equal to M/2. Then H+h, with h assuming integer values in the interval $-H < h \leqq (M+1)/2$, will be the numbers replacing the X-coordinates $x_1, \ldots, x_M$ for the purpose of fitting the polynomials. For the purpose of the fittings, all X-coordinates will be translated by **-H**, so that X-coordinates of the points to which the polynomials are fitted are taken to be the integers h, $-H \leqq h \leqq (M+1)/2$. Set

$$F(x) = \sum_{\iota=0}^{N-1} a_\iota x^\iota ,$$

and

$$G(x) = \sum_{\iota=0}^{N-1} b_\iota x^\iota .$$

Let C be the (MxN) matrix

$$C = [h_k^{\iota-1}],$$

where $h_k = -H+k$, $1 \leqq k \leqq M$, and $1 \leqq \iota \leqq N$. Let $Y = [y_1 \ldots y_M]^T$ and $Z = [z_1 \ldots z_M]^T$. Also let $A = [a_0 \ldots a_{N-1}]^T$ and $B = [b_0 \ldots b_{N-1}]^T$. Then the values of A and B for which the polynomials F and G give the best fit, in the sense of least squares, to the respective sets of pairs of numbers (h_k, y_k) and $h_k, z_k)$, $1 \leqq k \leqq M$, are given by

$$A = C^\ell Y$$

and

$$B = C^\ell Z.$$

Here, as in Section 2, $C^\ell = (C^T C)^{-1} C^T$.

It is clear that for fixed N and M, that the matrices C and C^ℓ are independent of the particular sequence of points $P_1, \ldots, P_M$, and therefore in an algorithm for computing the column vectors A and B at the various intervals along the scan lines, C^ℓ may be specified once and for all.

The polynomials F and G, with their coefficients determined as above, are used to obtain elevation estimates at points with integer X-coordinate as follows. Let m be the integer closest to the midpoint of the interval from x_1 to x_M. Then G(m-H) is the estimated elevation at the point (m,F(m-H)).

The first step of the array translocation is complete when all consecutive sets of M observation points on the epipolar scan lines have been processed in the above fashion to produce a new collection of points all having integer X-coordinates and estimated elevations.

The second step is a repetition of the first step, but now along lines parallel to the map coordinate system Y-axis and intercepting the map coordinate system X-axis in points with integer X-coordinates. The points to which the polynomials are fitted are the ones produced in the first step; the points at which the elevations are estimated are the ones with both coordinates integers. These are the points of the elevation grid.

This completes the general description of array translocation. Given below are some examples of the basic translocation fitting and estimation procedures for various values of N and M, accompanied by efficient algorithms for executing the computations.

4.2 Examples and Algorithms for Array Translocation

a) $N = 1, \quad M = 1$

For this case the elevation estimate at a point Q with integer X-coordinate m is the value of the elevation at the epipolar scanner observation point nearest to Q. The Y-coordinate of Q is the Y-coordinate of this observation point.

b) $N = 2, \quad M = 2$

This is ordinary linear interpolation.

c) $N = 2, \quad M = 3$

Here

$$C^{\ell} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix},$$

and Y and Z have size (2×1).

Then if $m' = m-H$,

$$F(m') = [1 \quad m']A$$
$$= [1 \quad m']C^{\ell}Y.$$

An algorithm for evaluation F at m' is as follows:

$$d_1 = (y_1 + y_2 + y_3)/3$$
$$d_2 = (y_3 - y_1)\, 2^{-1}$$
$$F(m') = d_1 + d_2 m'.$$

This requires only two multiplications by the constants 3^{-1} and 2^{-1}, and one multiplication $d_2 m'$. The function G is evaluated at m' by a similar algorithm with Y replaced by Z.

d) $N = 2, \quad M = 4$

For this case,

$$C^{\ell} = \frac{1}{6} \begin{bmatrix} 3 & 2 & 1 & 0 \\ -3 & -1 & 1 & 3 \end{bmatrix}$$

and Y and Z have size (4×1).

An algorithm to evaluate

$$F(m') = [1 \quad m'] \, C^{\ell} Y$$

is as follows:

$$d_1 = y_1 + y_2$$

$$d_2 = d_1 + d_1$$

$$d_3 = y_1 + y_3$$

$$d_4 = (d_1 + d_2)/6$$

$$d_5 = (y_4 - y_2)\, 2^{-1}$$

$$d_6 = d_4 + d_5$$

$$F(m') = d_5 + d_6 m'.$$

Even though there are now four observations, evaluation of each of the polynomials still only require two multiplications by the constants 2^{-1} and 6^{-1}, and the multiplication $d_6 m'$, which is the same number as is required in case (c) where $M = 3$. The fourth observation may cause some oversmoothing; however, including it improves the chances for the detection of gross errors. This may be done by forming $\sum_{i=1}^{4} z_i^2 - 4G(m')$, and if it is unexpectedly large, the residuals $z_i - G(x_i)$ may then be tested to determine the point P_i at which the gross error occurs.

e) $N = 3, \quad M = 5$

Here

$$C^{\ell} = \begin{bmatrix} -3/35 & 12/35 & 17/35 & 12/35 & -3/35 \\ -1/5 & -1/10 & 0 & 1/10 & 1/5 \\ 1/7 & -1/14 & -1/7 & 1/14 & 1/7 \end{bmatrix}$$

and Y and Z have size (5×1).

Instead of estimating the elevation at a single point, simultaneous estimation at three points will be considered in this example. These three points are $(m', F(m'))$ and also $(m'-1, F(m'-1))$ and $(m'+1, F(m'+1))$.

The algorithm for evaluation of G at $m'-1$, m' and $m'+1$ is as follows:

$$
\begin{aligned}
d_1 &= z_2 + z_4 \\
d_2 &= d_1 + d_1 \\
d_3 &= d_2 + d_2 \\
d_4 &= z_1 + z_5 \\
d_5 &= d_1 - z_3 \\
d_6 &= d_4 + d_4 \\
d_7 &= (3/35)(-d_4 + d_3 + (17/3)\, z_3) \\
d_8 &= (1/10)(z_4 - z_2 + z_5 - z_1 + z_5 - z_1) \\
d_9 &= (1/14)(d_6 - d_1).
\end{aligned}
$$

So far, only four multiplications, each with one factor fixed, are required. Then

$$
\begin{aligned}
e_1 &= d_9\, m' \\
e_2 &= d_8 + e_1 + e_1 \\
F(m') &= d_7 + m'(d_8 + e_1) \\
F(m'+1) &= F(m') + d_9 + e_2 \\
F(m'-1) &= F(m') + d_9 - e_2 .
\end{aligned}
$$

Two more scalar multiplications are required above.

This case has the advantage of not oversmoothing, as well as producing three elevation estimations at each iteration.

If a sparser elevation grid is wanted, e.g., only at points with even integer map coordinates, then the last part of the above algorithm may be easily modified for this as follows:

$$
\begin{aligned}
e_1 &= d_9 m' \\
e_2 &= d_8 + e_1 + e_1 \\
e_3 &= d_9 + d_9 \\
e_4 &= e_2 + e_2 \\
e_5 &= e_3 + e_3 \\
F(m') &= d_7 + m'(d_8 + e_1) \\
F(m'+2) &= F(m') + e_4 + e_5 \\
F(m'-2) &= F(m') + e_4 - e_5 .
\end{aligned}
$$

5. SUMMARY AND CONCLUSIONS

In the previous sections the principles of array algebra, as it applies to digital terrain representation, were put forth. Section 2 detailed the mathematical formalization for array regression in an N-dimensional Euclidean space. Section 3 put forth the algorithms for data compaction in an epipolar coordinate frame while Section 4 detailed the methods of array translocation as it would be applied to generating rectangularlyscaled terrain elevation.

During this study it was found that efficient direct array prediction algorithms could be derived and that the explicit formulation of the regression coefficients could be derived and that the explicit formulation of the regression coefficients could be avoided. Using these algorithms the array requirement for rectangularly spaced data was removed; thereby allowing boundaries of the area to assume arbitrary shapes. In delineating these methods several examples of direct prediction coefficients, or "k-vectors", were derived. In typical cases, the knowledge of ten (10) such coefficients replaces the data handling as has been conventionally required for explicit formulation of observation and normal equations, solution of normal equations and predictions using the solution.

The algorithms developed for translocation demonstrate that between 2 and 10 multiplications are required for each point manipulated.

In addition to analyzing the mathematical equations required for terrain compaction, the computational requirements were analyzed for both sequential and parallel processors. Based upon this analysis it appears that the array processor will provide a significant computational advantage.

As was previously stated two alternate methods; namely, compaction and translocation, were reviewed and mathematically described herein. Based upon our analysis it appears that these processes could have a significant impact on the

computational efficiency in the field of automated cartography. For this reason we recommend that these algorithms be implemented at the ETL facility and a series of comparative tests conducted using both the CDC 6400 and Staran computer systems. The exploitation of Staran, in conjunction with array regression, should form the cornerstone for a technology for efficient manipulation of massive volumes of digital terrain data. Increases in overall throughput of orders of magnitudes are totally within current state-of-the-art.

BIBLIOGRAPHY

1. D. B. Gennery, Image Enhancement, RCA Mathematical Services Technical Memorandum 5350-69-2 (1969).

2. K. Kraus and E. M. Mikhail, Linear Least-Squares Interpolation, Photogrammetric Engineering, 38(1972) 1016-1029.

3. U. Rauhala, Array Algebra with Applications in Photogrammetry and Geodesy, Fotogrammetriska Meddelanden, 6:6(1974) 1-144.

APPENDIX A

REVIEW OF THE R-MATRIX MULTIPLICATION PRINCIPLE OF ARRAY ALGEBRA

General Background and Definitions

This exposé is intended to explain some technical details pertinent to the R-matrix multiplication which forms the backbone of the so-called Array Algebra, treated in [Rauhala, 1976] and in many other publications by the same author. Considering the limited scope of this paper, the proofs of equivalence between certain least squares problems solved in a conventional way and the extremely efficient solutions of the same problems via array techniques will be omitted. We shall be satisfied by observing that the conventional (monolinear) least squares solution may be generalized to multilinear cases of array algebra provided the design matrix A in the monolinear formulation can be expressed as a tensor product of several matrices,

$$\underset{(M\times N)}{A} = \underset{(m_1\times n_1)}{A_1} \otimes \underset{(m_2\times n_2)}{A_2} \otimes \dots \otimes \underset{(m_i\times n_i)}{A_i} , \qquad (1)$$

$$\left.\begin{aligned} M &= m_1 \times m_2 \times \dots \times m_i , \\ N &= n_1 \times n_2 \times \dots \times n_i . \end{aligned}\right\} \qquad (1')$$

Here the symbol $\otimes$ represents a tensor product and we are in the presence of an i-dimensional case. The tensor product itself is defined in terms of two matrices as follows:

$$\underset{(p\times q)}{B} \otimes \underset{(r\times s)}{C} \equiv \underset{(pr\times qs)}{[B\,(C)_{k\ell}]} ,$$

i.e., each "entry" of the resulting matrix is the entry of the second matrix multiplied by the whole first matrix. From this definition follows e.g. the computational rule

$$\underset{(p\times q)}{B} \otimes \underset{(r\times s)}{C} \otimes \underset{(t\times u)}{D} \equiv \underset{(prt\times qsu)}{E} = \underset{(pr\times qs)}{(B\otimes C)} \otimes \underset{(t\times u)}{D} = \underset{(p\times q)}{B} \otimes \underset{(rt\times su)}{(C\otimes D)} .$$

We only mention that a formulation such as (1) is possible if the observations form an i-dimensional grid; throughout this paper, the pertinent weight matrices will be considered as unit matrices for simplicity.

The familiar observation equations in the mono-linear case read as follows:

$$\underset{(M \times 1)}{v} = \underset{(M \times N)(N \times 1)}{A\,x} - \underset{(M \times 1)}{\ell} \,,$$

where v is the residual vector, A is the design matrix, x is the vector of unknown parameters, and ℓ is the vector of observations (if the problem were not linear at the outset it would be linearized and a constant vector would be added to the right-hand side). The least squares solution may be written as

$$(x)_j = \sum_{r=1}^{M} (A^{\ell})_{jr} (\ell)_r \,, \qquad j = 1, 2, \ldots, N \,, \tag{2a}$$

where $(A^{\ell})_{jr}$ represents the entries of the matrix A^{ℓ} and $(x)_j$, $(\ell)_r$ represent the entries of the vectors x ,ℓ respectively;

$$\underset{(N \times M)}{A^{\ell}} \equiv \underset{(N \times N)}{(A^T A)^{-1}} \underset{(N \times M)}{A^T}$$

is called the l-inverse of the matrix A under the usual assumption of the nonsingular matrix of normal equations, $A^T A$. Written in the matrix form, equation (2a) becomes

$$\underset{(N \times 1)}{x} = \underset{(N \times M)(M \times 1)}{A^{\ell}\,\ell} \,. \tag{2b}$$

We shall now confine our attention to an array (multilinear) formulation associated with (1). We notice that ℓ will be written as an $(m_1 \times m_2 \times \ldots \times m_i)$ array L which stems from the gridded observational structure. In a similar fashion, x will be represented by an $(n_1 \times n_2 \times \ldots \times n_i)$ array X. Furthermore, the matrix A^{ℓ} can be shown to be the tensor product

$$\underset{(N \times M)}{A^{\ell}} = \underset{(n_1 \times m_1)}{A_1^{\ell}} \otimes \underset{(n_2 \times m_2)}{A_2^{\ell}} \otimes \ldots \otimes \underset{(n_i \times m_i)}{A_i^{\ell}} \,. \tag{2c}$$

From various references on the subject of array algebra we observe that the solution equivalent to (2a), (2b) is the following:

$$\left.\begin{aligned} &(X)_{j_1 j_2 \cdots j_i} = \sum_{r_1=1}^{m_1} (A_1^{\ell})_{j_1 r_1} \sum_{r_2=1}^{m_2} (A_2^{\ell})_{j_2 r_2} \cdots \sum_{r_i=1}^{m_i} (A_i^{\ell})_{j_i r_i} (L)_{r_1 r_2 \cdots r_i}\,, \\ &\qquad j_1 = 1, 2, \ldots, n_1 \\ &\qquad j_2 = 1, 2, \ldots, n_2 \\ &\qquad \vdots \\ &\qquad j_i = 1, 2, \ldots, n_i\,. \end{aligned}\right\} \quad (3a)$$

In a compact form, this may be written as

$$\underset{(n_1 \times n_2 \times \cdots \times n_i)}{X} = \underset{(n_1 \times m_1)}{(A_1^{\ell})^1} \; \underset{(n_2 \times m_2)}{(A_2^{\ell})^2} \cdots \underset{(n_i \times m_i)}{(A_i^{\ell})^i} \; \underset{(m_1 \times m_2 \times \cdots \times m_i)}{L}\,. \quad (3b)$$

An explanation of this notation convention is in order. The symbols X and L represent i-dimensional arrays, while A_j^{ℓ}, j=1, 2,..., i, are matrices (two dimensional arrays). The subscript simply identifies the matrix, i.e., it has a role of a "name". Thus instead of A_1, A_2, A_3, etc., we could have written A, B, C, etc.; since a matrix A_j is assumed to have the dimensions $(m_j \times n_j)$, its l-inverse matrix A_j^{ℓ} has the dimensions $(n_j \times m_j)$. On the other hand, the superscript "k" of a matrix indicates that its second index corresponds to the k-th index of L. For instance, the matrix A_i^{ℓ} has been assigned the superscript "i"; this means that the second index of the elements in A_i^{ℓ} must be the same as the i-th index of the elements in L. Equation (3a) demonstrates this rule and shows that the index in question is r_i. Consequently, the order of the second dimension of A_i^{ℓ} in (3b) is the same as that of the i-th dimension of L, namely m_i. In analogy to the conformability requirement (the number of columns in a first matrix equals the number of rows in a second matrix) associated with the conventional matrix multiplications, we

may now speak of conformable arrays associated with the R-matrix multiplications. The right-hand side of (3b) is a typical example of i successive R-matrix multiplications in a convenient, "short-hand" notation. A "long-hand" equivalent of (3b) is of course equation (3a) in which $j_1, j_2, \ldots, j_i$ are the "free" indeces. It is worth mentioning that the order in which the matrices are arranged for the R-matrix multiplications is inconsequential provided the superscripts are not interchanged; equation (3b) could thus be written e.g. as

$$X = (A_i^\ell)^i (A_1^\ell)^1 \ldots (A_2^\ell)^2 L .$$

This is apparent from (3a) in which the summations may be performed in an arbitrary order.

We shall next illustrate a few special cases of R-matrix multiplications. If the space is one-dimensional (i=1), we have essentially

$$\underset{(n_1 \times 1)}{X} = \underset{(n_1 \times m_1)}{(A_1^\ell)^1} \underset{(m_1 \times 1)}{L} \equiv A_1^\ell L ,$$

which corresponds to (2b). The two dimensional case may be portrayed as

$$\underset{(n_1 \times n_2)}{X} = \underset{(n_1 \times m_1)}{(A_1^\ell)^1} \underset{(n_2 \times m_2)}{(A_2^\ell)^2} \underset{(m_1 \times m_2)}{L} \equiv \underset{(n_1 \times m_1)}{A_1^\ell} \underset{(m_1 \times m_2)}{L} \underset{(m_2 \times n_2)}{A_2^{\ell T}} ,$$

where $A_2^{\ell T} \equiv (A_2^\ell)^T$. That it is so becomes apparent from the "long-hand" form, namely

$$(X)_{j_1 j_2} = \sum_{r_1=1}^{m_1} (A_1^\ell)_{j_1 r_1} \sum_{r_2=1}^{m_2} (A_2^\ell)_{j_2 r_2} (L)_{r_1 r_2} \equiv \sum_{r_1=1}^{m_1} \sum_{r_2=1}^{m_2} (A_1^\ell)_{j_1 r_1} (L)_{r_1 r_2} (A_2^{\ell T})_{r_2 j_2} ,$$

where $(A_2^\ell)_{j_2 r_2}$ has been written as $(A_2^{\ell T})_{r_2 j_2}$. We can thus assert that the superscript "1" corresponds to the left-hand side multiplication and the

superscript "2" corresponds to the transposition and the right-hand side multiplication. Unfortunately, there is no equivalent for the superscript "3", etc., so that somewhat absurd names such as "back-hand" side or "beyond-hand" side multiplication would have to be invented if we wanted to illustrate, at least intuitively, some operations in terms of the conventional matrices. One might thus be tempted to visualize the R-matrix multiplications in three or i dimensions as follows:

$$\underset{(n_1 \times n_2 \times n_3)}{X} = \underset{(n_1 \times m_1)}{(A_1^\ell)^1} \; \underset{(n_2 \times m_2)}{(A_2^\ell)^2} \; \underset{(n_3 \times m_3)}{(A_3^\ell)^3} \; \underset{(m_1 \times m_2 \times m_3)}{L} \equiv A_1^\ell \overset{A_3^{\ell T}}{L} A_2^{\ell T} \quad ,$$

$$X = (A_1^\ell)^1 (A_2^\ell)^2 (A_3^\ell)^3 \ldots (A_i^\ell)^i L \equiv A_1^\ell \overset{A_3^{\ell T} \ldots A^{\ell T}}{L} A_2^{\ell T} \quad .$$

However, it should be sufficient to work solely in terms of the original definitions (3a) and (3b). We shall next present some practical considerations in handling equation (3a) and shall determine the number of significant computer operations (scalar multiplications) involved.

The form of equation (3a) suggests that each summation sign represents a "do-loop" in regular FORTRAN programming. The last do-loop may then be imagined to result in an array Z whose elements are

$$(Z)_{r_1 r_2 \cdots r_{i-1} j_i} = \sum_{r_i=1}^{m_i} (A_i^\ell)_{j_i r_i} (L)_{r_1 r_2 \cdots r_{i-1} r_i} \quad . \tag{3c}$$

In computing the (one) element of Z depicted in (3c), m_i multiplications must be performed with regard to the index r_i which is then replaced by the free index j_i. To fill-in the elements of Z for one value of j_i, the summation in (3c) which was seen to require m_i multiplications must be carried out $m_1 \times m_2 \times \ldots \times m_{i-1}$ times (m_1 times due to the index r_1, m_2 times due to r_2, etc.). Finally, such operations must be repreated for each j_i, i.e. n_i times, in order to complete the array Z. This step has accordingly

involved

$$m_i \times (m_1 \times m_2 \times \dots \times m_{i-1}) \times n_i$$

multiplications. In analogy to (3a) and (3b) we may now write

$$(X)_{j_1 j_2 \cdots j_i} = \sum_{r_1=1}^{m_1} (A_1^\ell)_{j_1 r_1} \sum_{r_2=1}^{m_2} (A_2^\ell)_{j_2 r_2} \cdots \sum_{r_{i-1}=1}^{m_{i-1}} (A_{i-1}^\ell)_{j_{i-1} r_{i-1}} (Z)_{r_1 r_2 \cdots r_{i-1} j_i} , \quad (3d)$$

$$\underset{(n_1 \times n_2 \times \dots \times n_i)}{X} = \underset{(n_1 \times m_1)}{(A_1^\ell)^1} \; \underset{(n_2 \times m_2)}{(A_2^\ell)^2} \cdots \underset{(n_{i-1} \times m_{i-1})}{(A_{i-1}^\ell)^{i-1}} \; \underset{(m_1 \times m_2 \times \dots m_{i-1} \times n_i)}{Z} .$$

A similar argument to the above may be repeated with respect to the last do-loop in (3d) giving rise to the elements $(Y)_{r_1 r_2 \cdots r_{i-2} j_{i-1} j_i}$ of $\underset{(m_1 \times m_2 \times \dots \times m_{i-2} \times n_{i-1} \times n_i)}{Y}$, which now involves

$$m_{i-1} \times (m_1 \times m_2 \times \dots \times m_{i-2} \times n_i) \times n_{i-1}$$

multiplications. The number of multiplications that occur in the remainder of the i steps may be similarly expressed as

$$m_{i-2} \times (m_1 \times m_2 \times \dots \times m_{i-3} \times n_{i-1} \times n_i) \times n_{i-2}$$

$$\vdots$$

$$m_1 \times (n_2 \times n_3 \times \dots \times n_i) \times n_1 .$$

It may be appreciated that although the result is invariant with respect to the order in which the array multiplications are carried out, the total number of scalar multiplications is not.

In order to facilitate the upcoming comparisons in terms of scalar multiplications (and thus in terms of the computer time), the following simplifications are made:

$$m_1 = m_2 = \dots = m_i = m , \quad (4a)$$

$$n_1 = n_2 = \dots = n_i = n ; \quad (4b)$$

we have

$$m = kn \tag{5a}$$

where necessarily

$$k \geq 1 \ . \tag{5b}$$

The number of scalar multiplications associated with the R-matrix multiplications as described in the preceding paragraph is now

$$\left.\begin{aligned} m \times (m^{i-1}) \times n &= n^{i+1} k^{i} \\ m \times (m^{i-2} n) \times n &= n^{i+1} k^{i-1} \\ m \times (m^{i-3} n^2) \times n &= n^{i+1} k^{i-2} \\ &\vdots \\ m \times (n^{i-1}) \times n &= n^{i+1} k \ . \end{aligned}\right\} \tag{6}$$

The total is

$$i\, n^{i+1} \qquad \ldots \text{ if } \quad k = 1 \ , \tag{7a}$$

$$n^{i+1} k \, \frac{k^{i} - 1}{k - 1} \ \cdots \text{ if } \quad k > 1 \ . \tag{7b}$$

The formation of each of $\underset{(n \times m)}{A_j^{\ell}}$ from $\underset{(m \times n)}{A_j}$ requires approximately only $(2k+1)n^3$ scalar multiplications ($mn^2 = kn^3$ to form $A_j^T A_j$, approximately n^3 to invert this positive-definite matrix, and kn^3 to post-multiply this new matrix by A_j^T). The total in an i-dimensional problem is thus $i(2k+1)\,n^3$. If $i > 2$, this number is at least one order of magnitude smaller than the number of scalar multiplications associated with the subsequent R-matrix multiplications as evidenced by (7). Under such circumstances, the computer time required to form all of A_j^{ℓ} may be considered negligible.

Efficiency of the Array Technique

In cases where the design matrix has the structure exhibited in (1), the conventional procedure of forming the normal equations and solving for x as in (2b), i.e.

$$\underset{(N\times 1)}{x} = \underset{(N\times N)}{(A^T A)^{-1}} \underset{(N\times M)}{A^T} \underset{(M\times 1)}{\ell}$$

would be extremely uneconomical. In particular, the inversion of A^TA would necessitate approximately N^3 scalar multiplications. This inversion would consume by far the greatest amount of computer time. The formation of $A^T\ell$, by comparison, would require only NM scalar multiplications (although in general $M > N$, M is of the same order of magnitude as N) and the pre-multiplication of this vector by the above inverse matrix would necessitate only N^2 additional scalar multiplications. We have not mentioned the original construction of the matrix A^T in which slightly more than NM multiplications would be performed. In any event, the number of significant scalar multiplications determining the computer run time in this "brute force" approach would be N^3 and the operation directly responsible for it would be the matrix inversion of A^TA. Before proceeding any further we emphasize that the monolinear solution could be carried out much more efficiently through the use of the property (2c). If we set for simplicity

$$n_1 = n_2 = \ldots = n_i = n \quad ,$$

the "brute force" approach would involve one inversion of an $(n^i \times n^i)$ matrix while a "refined" monolinear approach will be seen to necessitate i inversions of small $(n \times n)$ matrices. Thus if n = 10, we would have e.g.

$i = 2$, $N = n^i = 100$ parameters; (100×100) inverse versus two (10×10) inverses

$i = 3$, $N = n^i = 1000$ parameters; (1000×1000) inverse versus three (10×10) inverses

The inversions of $(n \cdot n)$ matrices account for a negligible amount of the total computer time. On the other hand, the inversion of an $(N \times N)$ matrix in the "brute force" approach is in general totally prohibitive; this would not be substantially altered even if an inverse multiplication of the vector $A^T \ell$ by the matrix $A^T A$ were performed without explicitly inverting $A^T A$. The total number of scalar multiplications, although reduced, would still be proportional to N^3. The associated computer storage problems, not treated in this paper, would be equally formidable.

For the reasons exposed above, only the "refined" monolinear approach will be now considered and later compared with the array solution for efficiency. When the tensor product property (2c) is utilized, from (2b) and (2c) we write

$$x = (A_1^\ell \otimes A_2^\ell \otimes \ldots \otimes A_i^\ell) \ell . \qquad (8)$$

The formation of $A_1^\ell \otimes A_2^\ell \otimes \ldots \otimes A_i^\ell$ requires

$$n_1 m_1 \times n_2 m_2 + n_1 m_1 \times n_2 m_2 \times n_3 m_3 + \ldots + n_1 m_1 \times n_2 m_2 \times \ldots \times n_i m_i \qquad (9)$$

scalar multiplication, all of A_j^ℓ having been computed beforehand. Due to (1'), the last term in (9) is MN and the next largest term is smaller by at least two orders of magnitude. Since the number of scalar multiplications needed to form $A^\ell \ell$ in (8) is also MN, their total number is then 2MN.

Instead of (8), one might consider a mathematically equivalent but formally slightly different equation, namely

$$x = [(A_1^T A_1)^{-1} \otimes (A_2^T A_2)^{-1} \otimes \ldots \otimes (A_i^T A_i)^{-1}] A^T \ell .$$

However, in this formulation substantially more scalar multiplications are needed. Already the vector $A^T \ell$ involves 2MN of them since

$$A^T = A_1^T \otimes A_2^T \otimes \ldots \otimes A_i^T$$

is in this respect equivalent to A^ℓ; its subsequent post-multiplication by ℓ

necessitates another MN scalar multiplications. Furthermore, the construction of the matrix between the brackets in the above equation accounts for N^2 multiplications as does the final multiplication by $A^T\ell$. For these reasons, the most advantageous route pursued in the refined monolinear formulation is that of equation (8).

We now adopt the same simplifications as considered earlier in the equations (4) and (5); it follows that

$$M = m^i = k^i n^i \, , \tag{10a}$$

$$N = n^i \, . \tag{10b}$$

The total number of scalar multiplications associated with (8) is thus

$$2MN = 2k^i n^{2i} \, . \tag{11}$$

The number of multiplications used in forming all of A_j^ℓ was seen following (7) to be $i(2k+1)\,n^3$. This number is at least one order of magnitude smaller than (11) even if $i = 2$ and the formation of the matrices A_j^ℓ may accordingly be neglected insofar as the computer time is concerned. We shall next compare the above most favorable monolinear formulation with the array approach for efficiency. As a matter of interest we mention that if $k = 1$ (i.e. $n=m$, $N = M$), this monolinear formulation exhibits the structure related to the Fast Fourier Transform.

We have seen that both the refined monolinear formulation and the array approach require the formation of the A_j^ℓ matrices. In the former case, these operations are negligible compared to the total computer burden for any $i \geq 2$ while in the latter case, their impact can be ignored only if $i > 2$ (for $i = 2$, the number of pertinent scalar multiplications would have the same order of magnitude as in the subsequent R-matrix multiplications).

Accordingly, the "savings ratio" that will be determined in conjunction with the R-matrix multiplications will represent also the "total savings ratio" only for $i > 2$. If $i = 2$, the total savings ratio of the array solution versus the refined monolinear solution would be somewhat lower, although its order of magnitude would not be altered. The savings ratio (R) that concerns us is given by the expression in (11) divided by the number in (7a) or (7b). We thus have

$$R = (2/i)\, n^{i-1} \qquad \cdots \text{ if } k = 1 \ , \qquad (12a)$$

$$R = 2n^{i-1}(k^i - k^{i-1})/(k^i - 1) \ \ldots \text{ if } k > 1 \ . \qquad (12b)$$

For a large value of k in (12b), i.e. for many more observations than parameters, it follows that $R \to 2n^{i-1}$. For example, if $i = 3$ and $n = 100$ resulting in $N = 1000000$ parameters, the (total) savings ratio would be $R \to 20000$. However, a resonable value for k in practice is $k = 2$; one can imagine that every second point in each dimension of the grid is a "node point" where the value of some function is to be determined from the least squares adjustment.

We conclude this section with a few examples to illustrate the savings one may achieve when utilizing the array techniques :

$i = 2$, $n = 10$, $N = 100$ parameters...	$k = 1$,	$R = 10.0$
	$k = 2$,	$R = 13.3$
	$k = 4$,	$R = 16.0$
	$k = 8$,	$R = 17.8$
	k large	$R \to 20.0$
$i = 3$, $n = 10$, $N = 1000$ parameters...	$k = 1$,	$R = 67$
	$k = 2$,	$R = 114$
	$k = 4$,	$R = 152$
	$k = 8$,	$R = 175$
	k large	$R \to 200$.

In this last group ($i = 3$), R has the role of the total savings ratio. The number $n = 10$ would of course be too small in practice. A quite realistic example in two dimensions might be the following:

$i = 2$, $n = 100$, $N = 10000$ parameters... $k = 2$, $R = 133$.

It is not difficult to realize that if the "brute force" approach were used in the cases that concern us (gridded observations), the savings ratio would be several orders of magnitude higher than the above R. In particular, it would be multiplied by

$$(N^3 + 2MN + N^2)/2MN \ ,$$

i.e. approximately by

$$N^2/2M = N/2k^i$$

if k is not excessively large. In practice N (total number of parameters) could range from several hundred to several million. This thought should merely warn the "problem inventor" that extremely significant computer savings may fail to materialize if enough care is not taken at the level of the problem design and formulation.

Future Savings Possibilities

We assume that an ideal parallel processor capable of performing a sufficient number of scalar multiplications simultaneously will become available in the future. The economies achieved in conjunction with the array techniques could then be greatly magnified. In particular, the parallel processor could perform the multiplications in (3c) involving some or all $r_1, r_2, \ldots r_i$ simultaneously. In considering for our purpose the computer time requirements in the latter case, $(L)_{r_1 r_2 \ldots r_i}$ would act as a vector with a simple index. For the forthcoming comparisons we again adopt the simplifications (4a) and (4b). At each of the i steps, the number of multiplications indicated between the parentheses in the expression (6) would be now performed simultaneously with the number of multiplications (m) factored before the parentheses. The total number to be considered in this context is thus

$$imn = ikn^2 \, . \tag{13}$$

Upon consulting (7a) and (7b), we deduce that the additional savings ratio ($\bar{R}$) due to the parallel processor is

$$\bar{R} = n^{i-1} \qquad \ldots \text{ if } k = 1 \quad , \tag{14a}$$

$$\bar{R} = (n^{i-1}/i) * (k^i - 1)/(k-1) \quad \ldots \text{ if } k > 1 \quad . \tag{14b}$$

The pertinent savings ratio ($\tilde{R}$) between the refined monolinear solution and the array solution in conjunction with the parallel processor is

$$\tilde{R} = R\bar{R} \; . \tag{15}$$

Upon considering (12) and (14), for any $k \geq 1$ we have

$$\tilde{R} = (2/i) k^{i-1} n^{2(i-1)} \; .$$

This result is verified if the number $2k^i n^{2i}$ of (11) is divided by ikn^2 of (13).

We conclude by extending some of the examples of the previous section as follows (the values of $\tilde{R}$ are rounded):

$i = 2$, $n = 10$, $N = 100$ parameters... $k = 1$, $\bar{R} = 10$, $\tilde{R} = 100$
$k = 2$, $\bar{R} = 15$, $\tilde{R} = 200$
$k = 4$, $\bar{R} = 25$, $\tilde{R} = 400$

$i = 3$, $n = 10$, $N = 1000$ parameters... $k = 1$, $\bar{R} = 100$, $\tilde{R} = 6700$
$k = 2$, $\bar{R} = 233$, $\tilde{R} = 27000$
$k = 4$, $\bar{R} = 700$, $\tilde{R} = 107000$

$i = 2$, $n = 100$, $N = 10000$ parameters... $k = 2$, $\bar{R} = 150$, $\tilde{R} = 20000$.

As a closing remark we note that the idea of a parallel processor suits perfectly the R-matrix multiplications.

Practical Applications

The usefulness of the R-matrix multiplication technique for a whole class of least squares problems has been demonstrated at the level of the solution itself. The rule governing these multiplications has been explained following equation (3b), in which A_j^{ℓ} were the 1-inverses of the "original" matrices A_j. However, essentially the same rule could be applied and comparable savings could be realized with regard to the least squares predictions in which some "original" matrices would be involved. In either case the matrices attributed a superscript are called R-matrices. Practical applications associated with a special structure of these matrices will be briefly mentioned in the next paragraph.

In practice the R-matrices often exhibit a banded structure which can significantly accelerate the R-matrix multiplications. The banded structure is the basis of an extremely efficient multilinear prediction technique called array prediction. According to [Rauhala, 1976], this technique, when applicable, has the potential to surpass several "fast transforms", local polynomials, and other conventional signal processing and modeling techniques in many sciences. This is especially true for the automation of the typical "map" sciences like geodesy, photogrammetry, cartography, meteorology, oceanography, geology, image processing, remote sensing, and other sciences of the space age technology as indicated in the above reference.

REFERENCE

Rauhala, U.A., "A Review of Array Algebra". Paper published in Fotogrammetriska Meddelanden, 2:38, Department of Photogrammetry, Royal Institute of Technology, Stockholm, Sweden, 1976.